Using SAS to Bridge the Gap between Computer Science and Quantitative Methods

David R. Dolkart, American Hospital Association

Introduction
The degree of specialization within the fields of computer science and quantitative methods presents a gap in analyses. The concentration of computer scientists on efficiency of design and detailed documentation oftentimes roadblocks quantitative analysis. On the other hand, researchers' orientation toward quick results leads to inefficient program designs and scarce documentation of data sets and program operations. To avoid the extremes found in this interdisciplinary gap, a balance between efficiency, documentation, and turnaround time is crucial. This balance rests heavily on five factors: the underlying analysis characteristics; the nature of the project deadline; computer resource needs; staff requirements; and, the probability of repeating an analysis. Failure to fully assess the weight of each of these factors is a short cut to detours and dead ends.

Underlying Characteristics
The data processing segment of a research project involves one or more of the following operations:
1. Data Edits
2. Data Manipulation
3. Report Generation
4. Statistical Calculation
5. Graphing
Generally, a programming language—FORTRAN, COBOL, or PL/1, for example—or a package like SAS handles projects involving the first three operations. However, when edit, manipulation, or report operations dominate a project, the balance sways toward the efficiency found in a programming language. In projects where these five operations are equally important, the balance rests on a combination of a programming language and SAS. With few exceptions, projects involving extensive statistical calculations and graphing require the use of SAS alone.

Defining the Project Deadline
When external forces determine a project deadline (for example, a corporate officer requests an analysis to support a critical decision), snatching the detail of the quantitative analysis and data processing steps is essential. Under this time pressure, minimizing design and documentation paperwork frees up valuable time for quality control. When deadlines are flexible, there is more time to spend on documentation. The documentation level cannot dominate and, therefore, roadblock the analysis. But, by the same token, this documentation cannot be so scarce as to leave no trace of the data processing mechanics. Deadline flexibility also allows more room to maximize computer efficiency. That is, to minimize storage, memory, and run-time requirements.

Computer Resource Needs
The programming language and statistical package mixture depend on the computer resource needs of an application. These needs include disk storage, computer memory, and run-time. The resource needs of large data sets point the mixture toward the efficiency found in programming languages. In other applications, the amount of edits, manipulation, and computation are the dominant factors in deciding resource requirements; and, therefore, in the programming language-statistical package mixture decision.

The Role of Staff Requirements
In some research analyses, the size and expertise of the project staff predetermine the mix. For example, suppose one person comprises the staff. The programming background of that person determines whether SAS, FORTRAN, PL/1, or COBOL are elements of the software mixture. Flexibility in staff size and expertise, on the other hand, provides more opportunities to design and implement an efficient processing combination.

What is the Probability of Repeating an Analysis?
The ad hoc nature of some research applications imply that the probability of repeating an analysis is near zero. Therefore, the development costs necessary to minimize run-time may outweigh total cost savings—regardless of the data set size. As the probability of repetition increases, the cost savings from minimizing run-times begins to offset development costs. In some instances, time constraints indicate the use of a statistical package alone—even though the probability of repetition is high. After meeting the initial deadline, more time exists for using a programming language to optimize efficiency.

Striving for the Optimal Mix
Examining the five processing attributes of quantitative analyses provides the means to strive for an optimal programming language and statistical package equation. The emphasis here is on to strive for. Given the deadline and staff constraints found in many research projects, it is rare to "calculate" this optimal equation. Since precise calculation is unrealistic, researchers need a practical method to analyze their data processing requirements. The table below outlines such a method by defining general attributes which yield three processing mixtures. In the extreme situations—case 1 and case 3—mixture decisions are rarely unclear. For example, the computation of monthly price indexes requires the efficiency of a programming language. Whereas, the exploratory nature of model fitting requires the flexibility of a statistical package. The intermediate situation described by case 2 includes difficult applications which stretch into case 1 or 3. For example, selecting 1.5 million records, from 5 million records, for statistical analysis falls in the fringe, which indicates the use of a programming language alone (case 1) or in combination with SAS (case 2).
Review the Data Processing Steps

After formulating the software mixture, it is crucial to: review programs for efficiency; estimate the required computer resources; and, verify data edits, manipulations, and formulas with a test on a small subset (for example, 20-100 observations) of the data. A careful review of programs for efficiency double checks whether computer resource use is at a reasonable level. Testing all program operations on a small subset of data is an essential quality control measure. Tests also minimize costly reruns by providing valuable statistics for estimating storage, memory, and run-time requirements for the full data set.

A Mixture for 4.4 Million Records

Analyzing physicians' charge patterns for twelve medical procedures becomes tricky when there are 4.4 million records in the data set. Even though these observations comprise 1.5 million records (33.6 percent of the data), the twelve procedures appear "randomly" throughout the 4.4 million insurance claims. That is, sorting the data does not eliminate the need to read each record.

With no assistance from utilities or existing programs, but with sufficient time and staff, a programming language provides the best way to pull out physicians who performed any of these 12 procedures. With its data manipulation and statistical computation capabilities, SAS provides the means to achieve flexibility in this analysis. Therefore, a combination of a programming language and SAS formulate an optimal solution for this analysis.

With sufficient lead time, a programmer wrote a PL/1 program which processes over 29,000 records per second. At this rate, 1.1 million Medicare records and 0.4 million Blue Shield private business records came out of the program in slightly over 2.5 minutes.

Run-time statistics show that an IBM sort utility arranged the raw (that is, not in SAS data set form) Medicare records at a rate of 15,000 per second. In
comparison, SAS's PROC SORT arranged the private business records (that is, in SAS data set form) at a rate just exceeding 6,500 per second. This rate differential clearly indicates the run-time savings available from a careful outline which identified the data processing steps.

Medicare data had different codes for comparable private business procedures. Here, the outline revealed another significant savings. It was more efficient to transform the Medicare codes—to match the private business codes—after calculating means for each physician-procedure-geographic location combination (that is, PROC MEANS; by Physician Procedure Location). That is, it is cheaper to recode 16.5 thousand summarized Medicare observations. The Medicare records were then merged with 13.5 thousand Blue Shield records—by physician, procedure code, and location.

Careful planning, the right combination of programming skills, and sufficient lead time bridged the interdisciplinary gap by reducing 4.4 million claims to 30,000 physician specific observations with minimized run-times and maximized analysis flexibility.

Summary
A balance between efficient programs, documentation, and quick results leads to an optimal mix of a programming language and a statistical package. Formulating the ingredients in this software mixture rests on a careful analysis of five factors:

1. the underlying analysis characteristics;
2. the nature of the project deadline;
3. computer resource requirements;
4. staff requirements;
5. the probability of repeating the analysis.

Furthermore, a careful review of a project's purpose and this data processing mix provide the road map to avoid errors and expensive program reruns. Ultimately, this optimal programming language-statistical package formula provides the vehicle to bridge the gap between computer science and quantitative methods.

1Data set size is a relative concept which depends on computer capacity. To any computer running at peak capacity one million records comprise a large data set. As the load on this computer diminishes, these records shrink toward a medium-size data set.

2The cost of collecting the data used in this paper was funded by Blue Cross and Blue Shield Associations and the Health Care Financing Administration under Contract 500-78-001.

3For the purpose of this paper, records and claims are synonymous.

4Note that this processing rate applies to an AMDAHL 470/V8 configuration which processes over six million instructions per second.